Amendments to the Specification

Please replace the paragraph beginning at page 13, line 27, with the following rewritten paragraph:

As described above, the thread system handles the situation when a COM object returns a reference to another COM object. In such a case, the Java program does not explicitly instantiate the COM object to be returned. Rather, the COM object to be returned is instantiated and returned as a parameter of a method of another COM object. In one embodiment, the thread system keeps track of which thread returned the COM object (i.e., the returning thread). The COM object is returned by returning a pointer to the IUnknown interface of the COM object. When a COM object is returned, the thread system instantiates a wrapper object. The thread system then stores the thread identifier of the returning thread in the home thread field of the wrapper object for the returned object. Whenever a method of the returned COM object is invoked, the thread system intercepts the invocation and requests the home thread to access the COM object. Such an approach of always requesting the home thread is a conservative approach. In particular, if the returned object happens to be thread-safe, then there would be no need to request the home thread to invoke the method. Rather, the thread system could simply invoke the method from the current thread. However, the thread system may not be able to determine the thread type of the returned COM object. In particular, the thread system may not know the class identifier of the COM object and thus cannot retrieve the thread type from the registry. In another embodiment, the thread system makes use of a feature called free-threaded marshaling to identify if the COM object is thread-safe. According to the concept of free-threaded marshaling, when a thread-safe object is requested to be marshaled to another thread within the same process, a pointer directly to that object can be returned, rather than using the standard marshaling technique. The standard marshaling technique uses a proxy object and a stub object. Each time an object is marshaled using standard marshaling a new proxy object is instantiated. Thus, marshaled pointers to the same COM object will have different values, since they point to different proxy objects. However, pointers to the same COM object that are marshaled using free-threaded marshaling will have the same value, since they point to the COM object itself and not to a proxy object. When the thread system attempts to first invoke a method of a returned

COM object, the thread system requests the returning thread to marshal a pointer to the COM object to the current thread. The returning thread marshals the pointer according to the marshaling technique of the COM object. When the thread system receives the marshaled pointer, the thread system compares that pointer to the pointer to the IUnknown interface for that returned COM object stored in the wrapper object. If both pointers to the IUnknown interface are the same, then the COM object is thread-safe and the thread system can directly invoke the methods of the COM object from any thread. However, if the pointers are not the same, then standard marshaling created a new proxy object to which the marshaled pointer points and thus the COM object is not thread-safe. When the COM object is **not** thread-safe, the thread system requests the returning thread identified in the home thread field to access the COM object[[s]].